

Online Sum-Radii Clustering^{*}

Dimitris Fotakis¹ and Paraschos Koutris²

¹ School of Electrical and Computer Engineering,
National Technical University of Athens, 157 80 Athens, Greece.
fotakis@cs.ntua.gr

² Department of Informatics, University of Washington, U.S.A.
pkoutris@cs.washington.edu

Abstract. In Online Sum-Radii Clustering, n demand points arrive online and must be irrevocably assigned to a cluster upon arrival. The cost of each cluster is the sum of a fixed opening cost and its radius, and the objective is to minimize the total cost of the clusters opened by the algorithm. We show that the deterministic competitive ratio of Online Sum-Radii Clustering for metric spaces other than the line is $\Theta(\log n)$, where the upper bound follows from a primal-dual algorithm and holds for general metric spaces, and the lower bound is valid for ternary Hierarchically Well-Separated Trees (HSTs) and for the Euclidean plane. Combined with the results of (Csirik et al., MFCS 2010), this result demonstrates that the deterministic competitive ratio of Online Sum-Radii Clustering changes abruptly, from constant to logarithmic, when we move from the line to the plane. We also show that Online Sum-Radii Clustering in metric spaces induced by HSTs is closely related to the Parking Permit problem introduced by (Meyerson, FOCS 2005). Exploiting the relation to Parking Permit, we obtain a lower bound of $\Omega(\log \log n)$ on the randomized competitive ratio of Online Sum-Radii Clustering in tree metrics, and a randomized $O(2^d \sqrt{d} \log \log n)$ -competitive algorithm for the d -dimensional Euclidean space. Moreover, we present a simple and memoryless randomized $O(\log n)$ -competitive algorithm and a deterministic $O(\log \log n)$ -competitive fractional algorithm, which both work for general metric spaces.

Keywords: Algorithms and Data Structures, Online Algorithms, Competitive Analysis, Sum-Radii Clustering

1 Introduction

In clustering problems, we seek a partitioning of n demand points into k groups, or *clusters*, so that a given objective function, that depends on the distance between points in the same cluster, is minimized. Typical examples are the k -Center problem, where we minimize the maximum cluster diameter, the Sum- k -Radii problem, where we minimize the sum of cluster radii, and the k -Median problem, where we minimize the total distance of points to the nearest cluster center. These are fundamental problems in Computer Science, with many important applications, and have been extensively studied from an algorithmic viewpoint (see e.g. [20] and the references therein).

In this work, we study an online clustering problem closely related to Sum- k -Radii. In the on-line setting, the demand points arrive one-by-one and must be irrevocably assigned to a cluster upon arrival. We require that once formed, clusters cannot be merged, split, or have their center or radius changed. The goal is to open a few clusters with a small sum of radii. However, instead of requiring that at most k clusters open, which would lead to an unbounded competitive ratio, we follow [7,8] and consider a Facility-Location-like relaxation of Sum- k -Radii, called *Sum-Radii Clustering*. In Sum-Radii Clustering, the cost of each cluster is the sum of a fixed opening cost and its radius, and we seek to minimize the total cost of the clusters opened by the algorithm.

In addition to applications in clustering and data analysis, Sum-Radii Clustering has applications in the location of wireless base stations, such as sensors [8,9] and antennas [3,17]. In such problems,

^{*} Research partially supported by an NTUA Basic Research Grant (PEBE 2009).

we have to place some wireless base stations and setup their communication range so that some communication demands are satisfied and the total setup and operational cost is minimized. A standard assumption is that the setup cost is proportional to the number of stations installed, and the operational cost for each station is proportional to its range (or a low-degree polynomial of it).

Related Work. In the offline setting, the problem of Sum- k -Radii and the closely related problem of Sum- k -Diameters³ have been thoroughly studied from the viewpoint of exact and approximation algorithms. Sum- k -Radii is **NP**-hard even in metric spaces of constant doubling dimension [16]. Gibson et al. [15] proved that Sum- k -Radii in Euclidean spaces of constant dimension is polynomially solvable, and presented an $O(n^{\log \Delta \log n})$ -time exact algorithm for Sum- k -Radii in general metric spaces, where Δ is the diameter [16]. As for approximation algorithms, Doddi et al. [10] proved that it is **NP**-hard to approximate Sum- k -Diameters in general metric spaces within a factor less than 2, and gave a bicriteria algorithm that achieves a logarithmic approximation using $O(k)$ clusters. Subsequently, Charikar and Panigrahy [7] presented a primal-dual $(3.504 + \varepsilon)$ -approximation algorithm for Sum- k -Radii in general metric spaces, which uses as a building block a primal-dual 3-approximation algorithm for Sum-Radii Clustering. Biló et al. [3] considered a generalization of Sum- k -Radii, where the cost is the sum of the α -th power of the clusters radii, for $\alpha \geq 1$, and presented a polynomial-time approximation scheme for Euclidean spaces of constant dimension.

Charikar and Panigrahy [7] also considered the incremental version of Sum- k -Radii. Similarly to the online setting, an incremental algorithm receives the demands one-by-one and assigns them to a cluster upon arrival. However, an incremental algorithm can also merge any of its clusters at any time. They presented an $O(1)$ -competitive incremental algorithm for Sum- k -Radii that uses $O(k)$ clusters.

In the online setting, where cluster reconfiguration is not allowed, the Unit Covering and the Unit Clustering problems have received most of the attention. In both problems, the demand points arrive one-by-one and must be irrevocably assigned to unit-radius balls upon arrival, so that the number of balls used is minimized. The difference is that in Unit Covering, the center of each ball is fixed when the ball is first used, while in Unit Clustering, there is no fixed center and a ball may shift and cover more demands. Charikar et al. [6] proved an upper bound of $O(2^d d \log d)$ and a lower bound of $\Omega(\log d / \log \log \log d)$ on the deterministic competitive ratio of Unit Covering in d dimensions. The results of [6] imply a competitive ratio of 2 and 4 for Unit Covering on the line and the plane, respectively. The Unit Clustering problem was introduced by Chan and Zarrabi-Zadeh [5]. The deterministic competitive ratio of Unit Clustering on the line is at most $5/3$ [11] and no less than $8/5$ [12]. Unit Clustering has also been studied in d -dimensions with respect to the L_∞ norm, where the competitive ratio is at most $\frac{5}{6}2^d$, for any d , and no less than $13/6$, for $d \geq 2$ [11].

Departing from this line of work, Csirik et al. [8] studied the problem of online clustering to minimize the sum of the setup costs and the diameters of the clusters (CSDF). Motivated by the difference between Unit Covering and Unit Clustering, they considered three models, the strict, the intermediate, and the flexible one, depending on whether the center and the radius of a new cluster are fixed at its opening time. Csirik et al. only studied CSDF on the line and proved that its deterministic competitive ratio is $1 + \sqrt{2}$ for the strict and the intermediate model and $(1 + \sqrt{5})/2$ for the flexible model. Recently, Divéki and Imreh [9] studied online clustering in two dimensions to minimize the sum of the setup costs and the area of the clusters. They proved that the competitive ratio of this problem lies in $(2.22, 9]$ for the strict model and in $(1.56, 7]$ for the flexible model.

Contribution. Following [8], it is natural and interesting to study the online clustering problem of CSDF in metric spaces more general than the line. In fact, we consider the closely related problem of

³ These problems are closely related in the sense that a c -approximation/competitive algorithm for Sum- k -Radii implies a $2c$ -approximation/competitive algorithm for Sum- k -Diameters, and vice versa.

Online Sum-Radii Clustering (OnlSumRad), and give upper and lower bounds on its deterministic and randomized competitive ratio for general metric spaces and for the d -dimensional Euclidean space.

We restrict our attention to the strict model of [8], where the center and the radius of each new cluster are fixed at its opening time. To justify our choice, we show that a c -competitive algorithm for the strict model implies an $O(c)$ -competitive algorithm for the intermediate or the flexible model.

We show that the deterministic competitive ratio of OnlSumRad for metric spaces other than the line is $\Theta(\log n)$, where the upper bound follows from a primal-dual algorithm and holds for general metric spaces, and the lower bound is valid for ternary Hierarchically Well-Separated Trees (HSTs) and for the Euclidean plane. This result is particularly interesting because it demonstrates that the deterministic competitive ratio of OnlSumRad (and of CSDF) changes abruptly, from constant to logarithmic, when we move from the line to the plane. We note that this does not happen when the cost of each cluster is proportional to its area [9].

Another interesting finding is that OnlSumRad in metric spaces induced by HTSs is closely related to the Parking Permit problem introduced by Meyerson [19]. In Parking Permit, we cover a set of driving days by choosing among K permit types, each with a given cost and duration. The permit costs are concave, in the sense that the cost per day decreases with the duration. The algorithm is informed of the driving days in an online fashion, and irrevocably decides on the permits to purchase, so that all driving days are covered by a permit and the total cost is minimized. Meyerson [19] proved that the competitive ratio of Parking Permit is $\Theta(K)$ for deterministic and $\Theta(\log K)$ for randomized algorithms. We prove that OnlSumRad in HSTs is a generalization of Parking Permit. Combined with the randomized lower bound of [19], this implies a lower bound of $\Omega(\log \log n)$ on the randomized competitive ratio of OnlSumRad. Moreover, we show that, under some mild assumptions, a c -competitive algorithm for Parking Permit with K types implies a c -competitive algorithm for OnlSumRad in HSTs with K levels.

On the positive side, we give a reduction from OnlSumRad in the d -dimensional Euclidean space to OnlSumRad in HSTs with $K = O(\log n)$ levels which increases the competitive ratio by a factor of $O(2^d \sqrt{d})$. Using the competitive-ratio-preserving reduction from OnlSumRad in HSTs with K levels to ParkPermit with K types and the randomized algorithm of [19], we obtain a randomized $O(2^d \sqrt{d} \log \log n)$ -competitive algorithm for OnlSumRad in the d -dimensional Euclidean space. Moreover, we present a simple and memoryless randomized algorithm that achieves a competitive ratio of $O(\log n)$ for general metric spaces.

We conclude with a deterministic $O(\log \log n)$ -competitive online algorithm for the fractional version of OnlSumRad in general metric spaces. In the fractional version, we maintain fractions of open clusters that cumulatively cover all demands. The fractional algorithm is based on the primal-dual approach of [2,1], and generalizes the fractional algorithm of [19] for Parking Permit. We leave as an open problem the existence of a randomized rounding procedure that converts the fractional solution to an (integral) clustering of cost within a constant factor of the original cost. Such a rounding procedure would imply an (asymptotically optimal) randomized $O(\log \log n)$ -competitive algorithm for OnlSumRad in general metric spaces.

Other Related Work. At the conceptual level, OnlSumRad is related to the problem of Online Facility Location (see e.g. [18,14,13]). However, the two problems exhibit a quite different behavior with respect to their competitive ratio, since the competitive ratio of Online Facility Location is $\Theta(\frac{\log n}{\log \log n})$, even on the line, for both deterministic and randomized algorithms [14].

2 Notation, Problem Definition, and Preliminaries

Notation. We consider a metric space (M, d) , where M is the set of points and $d : M \times M \mapsto \mathbb{N}$ is the distance function, which is non-negative, symmetric and satisfies the triangle inequality. For a set of points $M' \subseteq M$, we let $\text{diam}(M') \equiv \max_{u, v \in M'} \{d(u, v)\}$ be the diameter and $\text{rad}(M') \equiv \min_{u \in M'} \max_{v \in M'} \{d(u, v)\}$ be the radius of M' . In a *tree metric*, the points correspond to the nodes of an edge-weighted tree and the distances are given by the tree's shortest path metric. For some $\alpha > 1$, a *Hierarchically α -Well-Separated Tree* (α -HST) is a complete rooted tree with lengths on its edges such that: (i) the distance of each leaf to its parent is 1, and (ii) on every path from a leaf to the root, the edge length increases by a factor of α on every level. Thus, the distance of any node v_k at level k to its children is α^{k-1} and the distance of v_k to the nearest leaf is $(\alpha^k - 1)/(\alpha - 1)$. We slightly abuse the notation and identify a tree (or an HST) with the metric space induced by it.

A *cluster* $C(p, r) \equiv \{v : d(p, v) \leq r\}$ is determined by its center p and its radius r , and consists of all points within a distance at most r to p . The cost of a cluster $C(p, r)$ is the sum of its opening cost f and its radius r .

Sum-Radii Clustering. In the offline version of Sum-Radii Clustering, the input consists of a metric space (M, d) , a cluster opening cost f , and a set $D = \{u_1, \dots, u_n\}$ of demand points in M . The goal is to find a collection of clusters $C(p_1, r_1), \dots, C(p_k, r_k)$ that cover all demand points in D and minimize the total cost, which is $\sum_{i=1}^k (f + r_i)$.

Online Sum-Radii Clustering. In the online setting, the demand points arrive one-by-one, in an online fashion, and must be irrevocably assigned to an open cluster upon arrival. Formally, the input to Online Sum-Radii Clustering (OnlSumRad) consists of the cluster opening cost f and a sequence u_1, \dots, u_n of (not necessarily distinct) demand points in an underlying metric space (M, d) . The goal is to maintain a set of clusters of minimum total cost that cover all demand points revealed so far.

In this work, we focus on the so-called Fixed-Cluster version of OnlSumRad, where the center and the radius of each new cluster are irrevocably fixed when the cluster opens. Thus, the online algorithm maintains a collection of clusters, which is initially empty. Upon arrival of a new demand u_j , if u_j is not covered by an open cluster, the algorithm opens a new cluster $C(p, r)$ that includes u_j , and assigns u_j to it. The algorithm incurs an irrevocable cost of $f + r$ for the new cluster $C(p, r)$.

Competitive Ratio. We evaluate the performance of the algorithms presented in this work using *competitive analysis* (see e.g. [4]). A (randomized) algorithm is c -competitive if for any sequence of demand points, its (expected) cost is at most c times the cost of the optimal solution for the corresponding offline Sum-Radii instance. We highlight that the (expected) cost of the algorithm is compared against the cost of an optimal offline algorithm that is aware of the entire demand sequence in advance and has no computational restrictions whatsoever.

Simplified Optimal. The following proposition, whose proof can be found in the Appendix, Section A.1, simplifies the structure of the optimal solution in the competitive analysis of our algorithms.

Proposition 1. *Let S be a feasible solution of an instance \mathcal{I} of OnlSumRad. Then, there is a feasible solution S' of \mathcal{I} with a cost of at most twice the cost of S , where each cluster has a radius of $2^k f$, for some integer $k \geq 0$.*

Other Versions of Online Sum-Radii Clustering. For completeness, we discuss two seemingly less restricted versions of OnlSumRad, corresponding to the intermediate and the flexible model in [8]. In both versions, the demands are irrevocably assigned to a cluster upon arrival. In the Fixed-Radius version, only the radius of a new cluster is fixed when the cluster opens. The algorithm incurs an irrevocable cost of $f + r$ for each new cluster C of radius r . Then, new demands can be assigned to C ,

provided that $\text{rad}(C) \leq r$. In the Flexible-Cluster version, a cluster C is a set of demands with neither a fixed center nor a fixed radius. The algorithm's cost for each cluster C is $f + \text{rad}(C)$, where $\text{rad}(C)$ may increase as new demands are added to C . Clearly, the Fixed-Cluster version is a restriction of the Fixed-Radius version, which, in turn, is a restriction of the Flexible-Cluster version. The following proposition, whose proof can be found in the Appendix, Section A.2, shows that from the viewpoint of competitive analysis, all the three versions are essentially equivalent.

Proposition 2. *A c -competitive algorithm for the Fixed-Radius (resp. Flexible-Cluster) version implies a $2c$ -competitive (resp. $10c$ -competitive) algorithm for the Fixed-Cluster version.*

Parking Permit. In Parking Permit (ParkPermit), we are given a schedule of days, some of which are marked as driving days, and K types of permits, where a permit of each type k , $k = 1, \dots, K$, has cost c_k and duration d_k . The goal is to purchase a set of permits of minimum total cost that cover all driving days. In the online setting, the driving days are presented one-by-one, and the algorithm irrevocably decides on the permits to purchase based on the driving days revealed so far.

Meyerson [19, Theorem 2.2] observed that by losing a constant factor in the competitive ratio, we can focus on the *interval version* of the problem, where each permit is available over specific time intervals (e.g. a weakly permit is valid from Monday to Sunday). Moreover, every day is covered by a single permit of each type k , and each permit of type $k \geq 2$ has d_k/d_{k-1} permits of type $k - 1$ embedded in it (see e.g. Fig. 1 for the structure of an interval instance).

An interesting feature of Meyerson's algorithms is that they are *time-sequence-independent*, in the sense that they apply, with the same competitive ratio, even if the order in which the driving days are revealed is different from their time order (e.g. the adversary may mark Sept. 23 as a driving day, before marking Aug. 6 as a driving day). In fact, Meyerson's algorithms exploit the hierarchical decomposition of the time axis into intervals, imposed by the permit structure, and not the relative time order between different days / subintervals.

3 Online Sum-Radii Clustering and Parking Permit

In this section, we establish a correspondence between OnlSumRad and ParkPermit. Specifically, we show that OnlSumRad in tree metrics and the interval version of ParkPermit are essentially equivalent problems. A few of our results either are directly based on this correspondence or exploit this correspondence so that they draw ideas from ParkPermit. We start with the following theorem, which shows that OnlSumRad in tree metrics is a generalization of the interval version of ParkPermit.

Theorem 1. *A c -competitive algorithm for Online Sum-Radii Clustering in HSTs with $K + 1$ levels implies a c -competitive algorithm for the interval version of Parking Permit with K permit types.*

Proof. Given an instance \mathcal{I} of the interval version of ParkPermit with K permit types, we construct an instance \mathcal{I}' of OnlSumRad in an HST with $K + 1$ levels such that any feasible solution of \mathcal{I} is mapped, in an online fashion, to a feasible solution of \mathcal{I}' of equal cost, and vice versa.

Let \mathcal{I} be an instance of the interval version of ParkPermit with K permit types of costs c_1, \dots, c_K and durations d_1, \dots, d_K . Without loss of generality, we assume that $c_1 = 1$ and that all days are covered by the permit of type K . Given the costs and the durations of the permits, we construct a tree T with appropriate edge lengths, which gives the metric space for \mathcal{I}' . The construction exploits the tree-like structure of the permits in the interval version of the problem (see also Fig. 1). Specifically, the tree T has $K + 1$ levels, where the leaves correspond to the days of \mathcal{I} 's schedule, and each node at level k , $1 \leq k \leq K$, corresponds to a permit of type k .

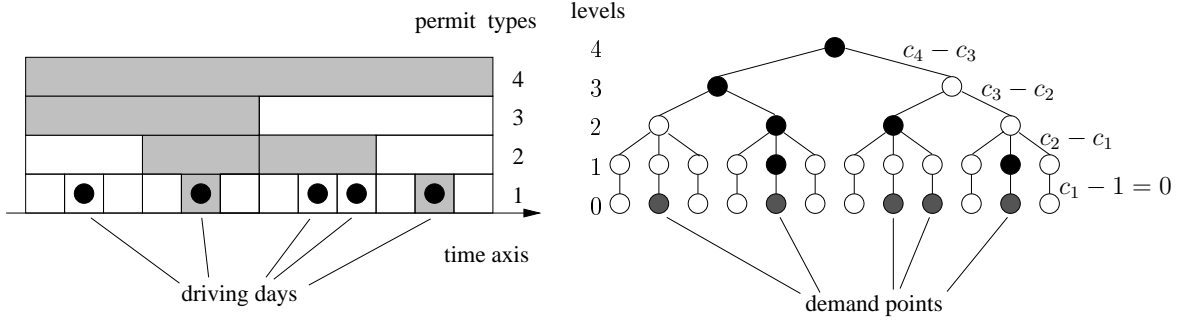


Fig. 1. An example of the reduction of Theorem 1. On the left, there is an instance of the interval version of ParkPermit. A feasible solution consists of the permits in grey. On the right, we depict the instance of OnlSumRad constructed in the proof of Theorem 1. The ParkPermit solution on left is mapped to a solution with clusters centered at each black node. The radius of each cluster is equal to the distance of its center to the nearest leaf.

Formally, the tree T has a leaf, at level 0, for each day in the schedule of \mathcal{I} . For each interval D_1 of d_1 days covered by a permit of type 1, there is a level-1 node v_1 in T whose children are the d_1 leaves corresponding to the days in D_1 . The distance of each level-1 node to its children is $c_1 - 1 = 0$. Hence, opening a cluster $C(v_1, c_1 - 1)$ covers all leaves corresponding to the days in the interval D_1 . Similarly, for each interval D_k of d_k days covered by a permit of type k , $2 \leq k \leq K$, there is a node v_k at level k in T whose children are the d_k/d_{k-1} nodes at level $k-1$ corresponding to the permits of type $k-1$ embedded within the particular permit of type k . The distance of each level- k node to its children is $c_k - c_{k-1}$. Therefore, opening a cluster $C(v_k, c_k - 1)$ covers all all leaves corresponding to the days in the interval D_k . The cluster opening cost is $f = 1$. For each day t marked as a driving day in \mathcal{I} , there is a demand located at the leaf of T corresponding to t in \mathcal{I}' .

Based on the correspondence between a type- k permit and a cluster $C(v_k, c_k - 1)$ rooted at a level- k node v_k , we show in the Appendix, Section A.3, that any feasible solution of \mathcal{I} is mapped, in an online fashion, to a feasible solution of \mathcal{I}' of equal cost, and vice versa. \square

In the proof of Theorem 1, if the ParkPermit instance has $d_1 = 1$ and $c_k = 2^k$, for all $k = 1, \dots, K$, the tree T is essentially a 2-HST with K levels where all nodes at the same level k have d_k/d_{k-1} children. Thus, combined with Theorem 1, the following lemma shows that OnlSumRad in such tree metrics is essentially equivalent to the interval version of ParkPermit. The proof of Lemma 1 applies essentially the reverse reduction of Theorem 1, and is deferred to the Appendix, Section A.4.

Lemma 1. *A c -competitive time-sequence-independent algorithm for the interval version of ParkPermit with K permits implies a c -competitive algorithm for OnlSumRad in HSTs with K levels, where all nodes at the same level have the same number of children and all demands are located at the leaves.*

4 Lower Bounds on the Competitive Ratio of Online Sum-Radii Clustering

By Theorem 1, Online Sum-Radii Clustering in trees with $K + 1$ levels is a generalization of Parking Permit with K permit types. Therefore, the results of [19] imply a lower bound of $\Omega(K)$ (resp. $\Omega(\log K)$) on the deterministic (resp. randomized) competitive ratio of OnlSumRad in trees with K levels. However, a lower bound on the competitive ratio of OnlSumRad would rather be expressed in terms of the number of demands n , because not only there is no simple and natural way of defining the number of “levels” of a general metric space, but also for online clustering problems, the competitive ratio, if not constant, is typically stated as a function of n .

Going through the details of the proofs of Theorem 1 and of [19, Theorems 3.2 and 4.6], we can translate the lower bounds on the competitive ratio of ParkPermit, expressed as a function of K , into

equivalent lower bounds for OnlSumRad, expressed as a function of n . In fact, the proofs of [19, Theorems 3.2 and 4.6] require that the ratio d_k/d_{k-1} of the number of days covered by permits of type k and $k-1$ is $2K$. Thus, in the proof of Theorem 1, the tree T has $(2K)^K$ leaves, and the number of demands n is at most $(2K)^K$. Combining this with the lower bound of $\Omega(\log K)$ on the randomized competitive ratio of ParkPermit [19, Theorem 4.6], we obtain the following corollary:

Corollary 1. *The competitive ratio of any randomized online algorithm for Online Sum-Radii Clustering in tree metrics is $\Omega(\log \log n)$, where n is the number of demands.*

A Stronger Lower Bound on the Deterministic Competitive Ratio. This approach gives a lower bound of $\Omega(\frac{\log n}{\log \log n})$ on the deterministic competitive ratio of OnlSumRad. Using a ternary HST instead, we next obtain a stronger lower bound.

Theorem 2. *The competitive ratio of any deterministic online algorithm for Online Sum-Radii Clustering in tree metrics is $\Omega(\log n)$, where n is the number of demands.*

Proof. For simplicity, let us assume that n is an integral power of 3. For some constant $\alpha \in [2, 3)$, we consider an α -HST T of height $K = \log_3 n$ whose non-leaf nodes have 3 children each. The cluster opening cost is $f = 1$. Let A be any fixed deterministic algorithm. We consider a sequence of demands located at the leaves of T . More precisely, starting from the leftmost leaf and advancing towards the rightmost leaf, the next demand point in the sequence is located at the next leaf not covered by an open cluster of A . Since T has n leaves, A may cover all leaves of T before the arrival of n demand points. Then, the demand sequence is completed in an arbitrary way that does not increase the optimal cost. We let C_{OPT} be the optimal cost, and let C_A be the cost of A on this demand sequence.

We let $c_k = 1 + \sum_{\ell=0}^{k-1} \alpha^\ell$ denote the cost of a cluster centered at a level- k node v_k with radius equal to the distance of v_k to the nearest leaf. We observe that for any $k \geq 1$ and any $\alpha \geq 2$, $c_k \leq \alpha c_{k-1}$. We classify the clusters opened by A according to their cost. Specifically, we let L_k , $0 \leq k \leq K$, be the set of A 's clusters with cost in $[c_k, c_{k+1})$, and let $\ell_k = |L_k|$ be the number of such clusters. The key property is that a cluster in L_k can cover the demands of a subtree rooted at level at most k , but not higher. Therefore, we can assume that all A 's clusters in L_k are centered at a level- k node and have cost equal to c_k , and obtain a lower bound of $C_A \geq \sum_{k=0}^K \ell_k c_k$ on the algorithm's cost.

To derive an upper bound on the optimal cost in terms of C_A , we distinguish between good and bad active subtrees, depending on the size of the largest radius cluster with which A covers the demand points in them. Formally, a subtree T_k rooted at level k is *active* if there is a demand point located at some leaf of it. For an active subtree T_k , we let $C_{T_k}^{\max}$ denote the largest radius cluster opened by A when a new demand point in T_k arrives. Let j , $0 \leq j \leq K$, be such that $C_{T_k}^{\max} \in L_j$. Namely, $C_{T_k}^{\max}$ is centered at a level- j node v_j and covers the entire subtree rooted at v_j . If $j \geq k$, i.e. if $C_{T_k}^{\max}$ covers T_k entirely, we say that T_k is a *good* (active) subtree (for the algorithm A). If $j < k$, i.e. if $C_{T_k}^{\max}$ does not cover T_k entirely, we say that T_k is a *bad* (active) subtree (for A).

For each $k = 0, \dots, K$, we let g_k (resp. b_k) denote the number of good (resp. bad) active subtrees rooted at level k . To bound g_k from above, we observe that the last demand point of each good active subtree rooted at level k is covered by a new cluster of A rooted at a level $j \geq k$. Therefore, the number of good active subtrees rooted at level k is at most the number of clusters in $\cup_{j=k}^K L_j$. Formally, for each level $k \geq 0$, $g_k \leq \sum_{j=k}^K \ell_j$. To bound b_k from above, we first observe that each active leaf / demand point is a good active level-0 subtree, and thus $b_0 = 0$. For each level $k \geq 1$, we observe that if T_k is a bad subtree, then by the definition of the demand sequence, the 3 subtrees rooted at the children of T_k 's root are all active. Moreover, each of these subtrees is either a bad subtree rooted at

level $k - 1$, in which case it is counted in b_{k-1} , or a good subtree covered by a cluster in L_{k-1} , in which case it is counted in ℓ_{k-1} . Therefore, for each level $k \geq 1$, $3b_k \leq b_{k-1} + \ell_{k-1}$.

Using these bounds on g_k and b_k , we can bound from above the optimal cost in terms of C_A . To this end, the crucial observation is that we can obtain a feasible solution by opening a cluster of cost c_k centered at the root of every active subtree rooted at level k . Since the number of active subtrees rooted at level k is $b_k + g_k$, we obtain that for every $k \geq 0$, $C_{OPT} \leq c_k(b_k + g_k)$. Using the upper bound on g_k and summing up for $k = 0, \dots, K$, we have that $(K + 1)C_{OPT} \leq \sum_{k=0}^K c_k b_k + \sum_{k=0}^K c_k \sum_{j=k}^K \ell_j$.

Using that $c_k \leq \alpha^k$ and that $c_k \leq \alpha c_{k-1}$, which hold for all $\alpha \geq 2$, we bound the second term by:

$$\sum_{k=0}^K c_k \sum_{j=k}^K \ell_j = \sum_{k=0}^K \ell_k \sum_{j=0}^k c_j \leq \sum_{k=0}^K \ell_k \sum_{j=0}^k \alpha^j \leq \sum_{k=0}^K \ell_k c_{k+1} \leq \alpha \sum_{k=0}^K \ell_k c_k \leq \alpha C_A$$

To bound the first term, we use that for every level $k \geq 1$, $3b_k \leq b_{k-1} + \ell_{k-1}$ and $c_k \leq \alpha c_{k-1}$. Therefore, $(3/\alpha)b_k c_k \leq (b_{k-1} + \ell_{k-1})c_{k-1}$. Summing up for $k = 1, \dots, K$, we have that:

$$\frac{3}{\alpha} \sum_{k=1}^K b_k c_k \leq \sum_{k=1}^K b_{k-1} c_{k-1} + \sum_{k=1}^K \ell_{k-1} c_{k-1}$$

Using that $b_0 = 0$ and that $\alpha < 3$, we obtain that:

$$\frac{3}{\alpha} \sum_{k=0}^K b_k c_k \leq \sum_{k=0}^{K-1} b_k c_k + \sum_{k=0}^{K-1} \ell_k c_k \leq \sum_{k=0}^K b_k c_k + C_A \Rightarrow \sum_{k=0}^K b_k c_k \leq \frac{\alpha}{3-\alpha} C_A$$

Putting everything together, we conclude that for any $\alpha \in [2, 3)$, $(K + 1)C_{OPT} \leq (\alpha + \frac{\alpha}{3-\alpha})C_A$. Since $K = \log_3 n$, this implies the theorem. \square

A Lower Bound for Deterministic OnlSumRad on the Plane. Motivated by the fact that the deterministic competitive ratio of OnlSumRad on the line is constant [8], we study OnlSumRad in Euclidean spaces of small constant dimension. The following theorem uses a constant-distortion planar embedding of the ternary α -HST employed in the proof of Theorem 2, and establishes a lower bound of $\Omega(\log n)$ on the deterministic competitive ratio of OnlSumRad on the Euclidean plane.

Theorem 3. *The competitive ratio of any deterministic online algorithm for Online Sum-Radii Clustering on the Euclidean plane is $\Omega(\log n)$, where n is the number of demands.*

Proof sketch. Using a planar embedding of a ternary α -HST T with distortion $D_\alpha \leq \sqrt{2}\alpha/(\alpha - 2)$, we show that a c -competitive algorithm for OnlSumRad on the plane implies a $2cD_\alpha$ -competitive algorithm for HSTs. The details can be found in the Appendix, Section A.5. \square

5 An Optimal Primal-Dual Algorithm

In this section, we present a deterministic primal-dual algorithm for OnlSumRad in a general metric space (M, d) . Throughout this section, we assume that the optimal solution only consists of clusters with radius $2^k f$, where k is a non-negative integer (see also Proposition 1). For simplicity, we let $r_k = 2^k f$, if $k \geq 0$, and $r_k = 0$, if $k = -1$. Let $N = \mathbb{N} \cup \{-1\}$. Then, the following are a Linear Programming relaxation of OnlSumRad and its dual:

$$\begin{array}{ll}
\min & \sum_{(z,k) \in M \times N} x_{zk}(f + r_k) \\
\text{s.t.} & \sum_{(z,k): d(u_j, z) \leq r_k} x_{zk} \geq 1 \quad \forall u_j \\
& x_{zk} \geq 0 \quad \forall (z, k)
\end{array}
\qquad
\begin{array}{ll}
\max & \sum_{j=1}^n a_j \\
\text{s.t.} & \sum_{j: d(u_j, z) \leq r_k} a_j \leq f + r_k \quad \forall (z, k) \\
& a_j \geq 0 \quad \forall u_j
\end{array}$$

In the primal program, there is a variable x_{zk} for each point z of the metric space and each $k \in N$ that indicates the extent to which cluster $C(z, r_k)$ is open. The constraints require that each demand u_j is fractionally covered. If we require that $x_{zk} \in \{0, 1\}$ for all z, k , we obtain an Integer Programming formulation of OnlSumRad. In the dual program, there is a variable a_j for each demand u_j , and the constraints require that no potential cluster is “overpaid”.

The primal-dual algorithm for OnlSumRad, or PD-SumRad in short, maintains a collection of clusters that cover all the demands processed so far. When a new demand u_j , $j = 1, \dots, n$, arrives, if u_j is covered by an already open cluster C , PD-SumRad assigns u_j to C and sets u_j ’s dual variable a_j to 0. Otherwise, PD-SumRad sets a_j to f . This makes the dual constraint corresponding to $(u_j, -1)$ and possibly some other dual constraints tight. PD-SumRad finds the maximum $k \in N$ such that for some point $z \in M$, the dual constraint corresponding to (z, k) becomes tight due to a_j . Then, PD-SumRad opens a new cluster $C(z, 3r_k)$ and assigns u_j to it. The main result of this section is that:

Theorem 4. *The competitive ratio of PD-SumRad is $\Theta(\log n)$.*

The lower bound on the competitive ratio of PD-SumRad follows from Theorem 2. The proof of the upper bound consists of a pair of lemmas. In the Appendix, Section A.6, we show that the dual solution maintained by PD-SumRad is feasible. Thus the optimal cost for any demand sequence is at least the value of the dual solution maintained by PD-SumRad. The following lemma shows that the total cost of PD-SumRad is at most $O(\log n)$ times the value of its dual solution.

Lemma 2. *The cost of PD-SumRad is at most $3(2 + \log_2 n) \sum_{j=1}^n a_j$.*

Proof. We call a cluster $C(z, r_k)$ *tight* if the dual constraint corresponding to (z, k) is satisfied with equality. We observe that for any integer $k > \log_2 n$ and for all points z , $C(z, k)$ cannot become tight, because the lefthand-side of any dual constraint is at most nf . Therefore, we can restrict our attention to at most $2 + \log_2 n$ values of k .

Next, we show that for all $k = -1, 0, \dots, \lfloor \log_2 n \rfloor$, each demand u_j with $a_j > 0$ contributes to the opening cost of at most one cluster with radius $3r_k$. Namely, PD-SumRad opens at most one cluster $C(z, 3r_k)$ for which u_j belongs to the tight cluster $C(z, r_k)$. We prove this claim by contradiction. Let us assume that for some value of k , PD-SumRad opens two clusters $C_1 = C(z_1, 3r_k)$ and $C_2 = C(z_2, 3r_k)$ for which there is a demand u_j with $a_j > 0$ that belongs to both $C(z_1, r_k)$ and $C(z_2, r_k)$. Since PD-SumRad opens at most one new cluster when a new demand is processed, one of the clusters C_1, C_2 opens before the other. So, let us assume that C_1 opens before C_2 . This means that PD-SumRad opened C_1 in response to a demand $u_{j'}$, with $j' \leq j$, that was uncovered at its arrival time and made $C(z_1, r_k)$ tight. Then, any subsequent demand $u \in C(z_2, r_k)$ is covered by C_1 , because:

$$d(u, z_1) \leq d(u, u_j) + d(u_j, z_1) \leq 2r_k + r_k = 3r_k$$

The second inequality above holds because both u and u_j belong to $C(z_2, r_k)$ and u_j also belongs to $C(z_1, r_k)$. Therefore, after C_1 opens, there are no uncovered demands in $C(z_2, r_k)$ that can force PD-SumRad to open C_2 , a contradiction.

To conclude the proof of the lemma, we observe that when PD-SumRad opens a new cluster $C(z, 3r_k)$, the cluster $C(z, r_k)$ is tight. Hence, the total cost of $C(z, 3r_k)$ is at most $3 \sum_{u_j \in C(z, r_k)} a_j$. Therefore, the total cost of PD-SumRad is at most:

$$\begin{aligned} \sum_{(z,k): C(z, 3r_k) \text{ opens}} \sum_{u_j \in C(z, r_k)} 3a_j &= 3 \sum_{j=1}^n a_j |\{(z, k) : C(z, 3r_k) \text{ opens and } u_j \in C(z, r_k)\}| \\ &\leq 3(2 + \log_2 n) \sum_{j=1}^n a_j \end{aligned}$$

The inequality above holds because for each $k = -1, 0, \dots, \lfloor \log_2 n \rfloor$ and each demand u_j with $a_j > 0$, there is at most one pair (z, k) such that $C(z, 3r_k)$ opens and $u_j \in C(z, r_k)$. \square

6 Randomized and Fractional Algorithms

Throughout this section, we assume, essentially without loss of generality, that the optimal solution only consists of clusters of radius $2^k f$, where k is a non-negative integer.

A Simple Randomized Algorithm for General Metrics. We start with a simple randomized algorithm, or Simple-SumRad in short, with a logarithmic competitive ratio. Simple-SumRad is a *memoryless* algorithm, in the sense that it keeps in memory only the solution maintained by it, namely the centers and the radii of its clusters. For simplicity, we assume that n is an integral power of 2 and known to the algorithm in advance. This assumption can be easily removed by standard techniques (see also Section A.9, in the Appendix). When a new demand u_j arrives, if u_j is covered by an already open cluster C , Simple-SumRad assigns u_j to C . Otherwise, for each $k = 0, \dots, \log_2 n$, Simple-SumRad opens a new cluster $C(u_j, 2^k f)$ with probability 2^{-k} , and assigns u_j to the cluster $C(u_j, f)$, which opens with probability 1. In the Appendix, Section A.7, we show that:

Lemma 3. *Simple-SumRad achieves a competitive ratio of at most $2(4 + \log_2 n)$.*

A Randomized Algorithm for OnlSumRad in d Dimensions. Next, we present a reduction of OnlSumRad in the d -dimensional Euclidean space into OnlSumRad in HSTs where each internal node has 2^d children. The reduction increases the competitive ratio by a factor of $O(2^d \sqrt{d})$.

Lemma 4. *A $c(K)$ -competitive algorithm for OnlSumRad in HSTs with K levels, where all nodes at the same level have the same number of children and all demands are located at the leaves, implies a $O(2^d \sqrt{d} c(K))$ -competitive algorithm for OnlSumRad in the d -dimensional Euclidean space, with $K = O(\log n)$.*

Proof sketch. Given an instance \mathcal{I} of OnlSumRad in the d -dimensional Euclidean space, we construct an instance \mathcal{I}' of OnlSumRad in a 2-HST T with $K = O(\log n)$ levels. The HST T corresponds to a hierarchical decomposition \mathcal{D} of the d -dimensional space into hypercubes with $K + 1$ levels, where the hypercubes at level k have a side length of 2^k .

Using that any cluster of radius 2^{k-1} in \mathcal{I} intersects at most 2^d level- k hypercubes of the hierarchical decomposition \mathcal{D} , we show that any feasible solution of \mathcal{I} of cost c is mapped to a feasible solution of \mathcal{I}' of cost at most $O(2^d c)$. On the other hand, any cluster C_k of radius $2^k - 1$ in T corresponds to a single level- k hypercube h_k of \mathcal{D} . Thus, the demands covered by C_k in T can be covered by a single cluster of radius $\sqrt{d} \cdot 2^{k-1}$ in \mathcal{I} . Therefore, given a feasible solution S of \mathcal{I}' of cost c , opening the clusters in the d -dimensional Euclidean space that correspond to the clusters of S , we obtain a feasible solution of \mathcal{I} of cost at most $\sqrt{d} c$. A detailed proof can be found in the Appendix, Section A.8. \square

The following theorem follows by combining Lemma 4 with Lemma 1, and applying the randomized $O(\log K)$ -competitive algorithm of [19, Section 4.1] for the interval version of ParkPermit. We note that the proof of Lemma 4 assumes that the number of demands n is known in advance. This assumption can be easily removed by standard techniques (see also Section A.9, in the Appendix).

Theorem 5. *There is a randomized $O(2^d \sqrt{d} \log \log n)$ -competitive algorithm for Online Sum-Radii Clustering in the d -dimensional Euclidean space.*

A Fractional Algorithm for General Metrics. We conclude with a deterministic $O(\log \log n)$ -competitive algorithm for the fractional version of OnlSumRad in general metric spaces. The fractional algorithm is based on the primal-dual approach of [2,1], and is a generalization of the online algorithm for the fractional version of ParkPermit presented in [19, Section 4.1].

A fractional algorithm maintains, in an online fashion, a feasible solution to the Linear Programming relaxation of OnlSumRad. In the notation of Section 5, for each point-type pair (z, k) , the algorithm maintains a fraction x_{zk} , which denotes the extent to which the cluster $C(z, r_k)$ opens, and can only increase as new demands arrive. For each demand u_j , the fractions of the clusters covering u_j must sum up to at least 1, i.e. $\sum_{(z,k): u_j \in C(z, r_k)} x_{zk} \geq 1$. The total cost of the algorithm is $\sum_{(z,k)} x_{zk}(f + r_k)$. The competitive ratio is the worst-case ratio of the algorithm's cost to the cost of an optimal integral solution on the same demand sequence.

The Fractional Algorithm. For the fractional algorithm, or Frac-SumRad in short, we assume that the number of demands n is an integral power of 2 and known in advance. In the Appendix, Section A.9, we remove these assumptions, by losing a constant factor in the competitive ratio.

Frac-SumRad considers only $K + 1$ different types of clusters, where $K = \log_2 n$. For each $k = 1, \dots, K + 1$, we let $c_k = f + r_k$ denote the cost of a cluster $C(p, r_k)$ of type k . The algorithm considers only the demand locations as potential cluster centers. For convenience, for each demand u_j and for each k , we let x_{jk} be the extent to which the cluster $C(u_j, r_k)$ is open, with the understanding that $x_{jk} = 0$ before u_j arrives. Similarly, we let $F_{jk} = \sum_{(i,k): u_j \in C(u_i, r_k)} x_{ik}$ be the extent to which demand u_j is covered by clusters of type k , and let $F_j = \sum_k F_{jk}$ be the extent to which u_j is covered.

When a new demand $u_j, j = 1, \dots, n$, arrives, if $F_j \geq 1$, u_j is already covered. Otherwise, while $F_j < 1$, Frac-SumRad performs the following operation:

1. For every $k = 1, \dots, K + 1$, $x_{jk} \leftarrow x_{jk} + \frac{1}{c_k(K+1)}$
2. For every $k = 1, \dots, K + 1$ and every demand $u_i \in C(u_j, r_k)$, $x_{ik} \leftarrow x_{ik}(1 + \frac{1}{c_k})$

Competitive Analysis. Frac-SumRad maintains a (fractional) feasible solution in an online fashion. The proof of the following theorem extends the competitive analysis in [19, Section 4.1].

Theorem 6. *The competitive ratio of Frac-SumRad is $O(\log \log n)$.*

Proof. We first consider a single operation performed when a demand u_j arrives, and show that it increases the fractional cost by at most 2. Since an operation is performed, $F_j < 1$. The first step of the operation increases the fractional cost by $1/(K+1)$ for each cluster type. Hence, the total increase in the fractional cost is 1. The second step of the operation increases the fractional cost by:

$$\sum_{(i,k): u_i \in C(u_j, r_k)} x_{ik} = \sum_{(i,k): u_i \in C(u_j, r_k)} x_{ik} = \sum_{k=1}^{K+1} F_{jk} = F_j < 1$$

We next show that the number of operations performed by Frac-SumRad for the demands in an optimal cluster $C(p, r_k)$ of cost c_k is $O(c_{k+1} \log K)$. We let $F_{p(k+1)} = \sum_{j: u_j \in C(p, r_k)} x_{j(k+1)}$. Since

for any demand $u_j \in C(p, r_k)$, $C(u_j, r_{k+1})$ includes the entire $C(p, r_k)$, we have that $F_{j(k+1)} \geq F_{p(k+1)}$. Hence, as soon as $F_{p(k+1)} \geq 1$, every subsequent demand $u_j \in C(p, r_k)$ has $F_j \geq 1$ at its arrival time, and Frac-SumRad does not perform any operations due to demands u_j . Consequently, the total cost of Frac-SumRad for the demands in $C(p, r_k)$ can be bounded by the total increase in the fractional cost due to operations caused by demands in $C(p, r_k)$ arriving as long as $F_{p(k+1)} < 1$.

To bound the number of such operations, we observe that after the first c_{k+1} operations caused by demands in $C(p, r_k)$, $F_{p(k+1)}$ becomes at least $1/(K+1)$, due to the first step of these operations. For each subsequent operation caused by a demand in $C(p, r_k)$, all fractions $x_{j(k+1)}$, with $u_j \in C(p, r_k)$, increase by factor of $(1 + \frac{1}{c_{k+1}})$. Therefore, $F_{p(k+1)}$ increases by a factor of $(1 + \frac{1}{c_{k+1}})$. After $O(c_{k+1} \log K)$ such increases, $F_{p(k+1)}$ becomes at least 1, and Frac-SumRad does not perform any additional operations due to demands in $C(p, r_k)$ arriving afterwards.

Therefore, the total fractional cost of Frac-SumRad for the demands in an optimal cluster $C(p, r_k)$ of cost c_k is $O(c_{k+1} \log K)$. Then, the theorem follows from $c_{k+1} \leq 2c_k$ and $K = \log_2 n$. \square

References

1. N. Alon, B. Awerbuch, Y. Azar, N. Buchbinder, and J. Naor. A General Approach to Online Network Optimization Problems. *ACM Transactions on Algorithms*, 2(4):640–660, 2006.
2. N. Alon, B. Awerbuch, Y. Azar, N. Buchbinder, and J. Naor. The Online Set Cover Problem. *SIAM J. on Computing*, 39(2):361–370, 2009.
3. V. Biló, I. Caragiannis, C. Kaklamanis, and P. Kanellopoulos. Geometric Clustering to Minimize the Sum of Cluster Sizes. In *Proc. of the 13th European Symposium on Algorithms (ESA '05)*, volume 3669 of *LNCS*, pages 460–471, 2005.
4. A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.
5. T.M. Chan and H. Zarrabi-Zadeh. A Randomized Algorithm for Online Unit Clustering. *Theory of Computing Systems*, 45(3):486–496, 2009.
6. M. Charikar, C. Chekuri, T. Feder, and R. Motwani. Incremental Clustering and Dynamic Information Retrieval. *SIAM J. on Computing*, 33(6):1417–1440, 2004.
7. M. Charikar and R. Panigrahy. Clustering to Minimize the Sum of Cluster Diameters. *J. of Computer and System Sciences*, 68(2):417–441, 2004.
8. J. Csirik, L. Epstein, C. Imreh, and A. Levin. Online Clustering with Variable Sized Clusters. In *Proc. of the 35th Symposium on Mathematical Foundations of Computer Science (MFCS '10)*, volume 6281 of *LNCS*, pages 282–293, 2010.
9. G. Divéki and C. Imreh. An Online 2-Dimensional Clustering Problem with Variable Sized Clusters. *Submitted for publication*, 2011.
10. S. Doddi, M.V. Marathe, S.S. Ravi, D.S. Taylor, and P. Widmayer. Approximation Algorithms for Clustering to Minimize the Sum of Diameters. *Nordic J. Computing*, 7(3):185–203, 2000.
11. M.R. Ehmsen and K.S. Larsen. Better Bounds on Online Unit Clustering. In *Proc. of the 12th Scandinavian Symposium on Algorithm Theory (SWAT '10)*, volume 6139 of *LNCS*, pages 371–382, 2010.
12. L. Epstein and R. van Stee. On the Online Unit Clustering Problem. *ACM Transactions on Algorithms*, 7(1):7, 2010.
13. D. Fotakis. A Primal-Dual Algorithm for Online Non-Uniform Facility Location. *Journal of Discrete Algorithms*, 5(1):141–148, 2007.
14. D. Fotakis. On the Competitive Ratio for Online Facility Location. *Algorithmica*, 50(1):1–57, 2008.
15. M. Gibson, G. Kanade, E. Krohn, I.A. Pirwani, and K. Varadarajan. On Clustering to Minimize the Sum of Radii. In *Proc. of the 19th ACM-SIAM Symposium on Discrete Algorithms (SODA '08)*, pages 819–815, 2008.
16. M. Gibson, G. Kanade, E. Krohn, I.A. Pirwani, and K. Varadarajan. On Metric Clustering to Minimize the Sum of Radii. *Algorithmica*, 57:484–498, 2010.
17. N. Lev-Tov and D. Peleg. Polynomial Time Approximation Schemes for Base Station Coverage with Minimum Total Radii. *Computer Networks*, 47(4):489–501, 2005.
18. A. Meyerson. Online Facility Location. In *Proc. of the 42nd IEEE Symposium on Foundations of Computer Science (FOCS '01)*, pages 426–431, 2001.
19. A. Meyerson. The Parking Permit Problem. In *Proc. of the 46th IEEE Symposium on Foundations of Computer Science (FOCS '05)*, pages 274–284, 2005.
20. S.E. Schaeffer. Graph Clustering. *Computer Science Review*, 1:27–64, 2007.

A Appendix

A.1 The Proof of Proposition 1

For each cluster $C(v, r)$ of S , S' opens a cluster $C(v, 2^k f)$, where $k = \max\{\lceil \log_2(r/f) \rceil, 0\}$. Clearly, $C(v, 2^k f)$ covers all the demand points covered by $C(v, r)$, and thus S' is a feasible solution. As for the cost of S' , we next show that the cost of $C(v, 2^k f)$, that is $(1 + 2^k)f$, is most twice the cost of $C(v, r)$, that is $f + r$. If $r < f$, in which case $k = 0$, the cost of $C(v, 2^k f)$ is $2f$. If $r \geq f$, $2^k f \leq 2^{1+\log_2(r/f)} f = 2r$. Therefore, the cost of $C(v, 2^k f)$ is at most $f + 2r \leq 2(f + r)$. \square

A.2 The Proof of Proposition 2

We first assume a c -competitive algorithm A for the FIXED-RADIUS version. Based on A , we describe an algorithm A' for the FIXED-CLUSTER version that simulates the behavior of A and has a competitive ratio of at most $2c$. Whenever the algorithm A opens a new cluster C of radius r and covers a new demand u , the algorithm A' opens a new cluster $C' = C(u, 2r)$. The cost of C' is at most twice the cost of C . Moreover, since any subsequent demand u' assigned to C by A is at distance at most $2r$ to u , the new cluster C' also covers u' . Therefore, A' covers all demands with a total cost at most twice the total cost of A .

Next, we assume a c -competitive algorithm A for the FLEXIBLE-CLUSTER version. Based on A , we describe an algorithm A' for the FIXED-CLUSTER version that simulates the behavior of A and has a competitive ratio of at most $10c$.

Let u be a new demand assigned to a cluster C by the algorithm A . If C is a new cluster that includes only u , A' opens a new cluster $C(u, f)$ and assigns u to it. Otherwise, let \hat{u} be the demand in C arrived first. If u is covered by an open cluster of A' centered at \hat{u} , u is assigned to it. Otherwise, A' opens a new cluster $C(\hat{u}, 2^k f)$, where $k = \lceil \log_2(d(\hat{u}, u)/f) \rceil$, and assigns u to it.

We compare the total cost of the algorithms A and A' for covering the demands in cluster C just after the assignment of u . The cost of A is at least $f + \text{diam}(C)/2$. If $\text{diam}(C) \leq f$, all demands in C are within a distance of f to C 's first demand \hat{u} , and are assigned to the cluster $C(\hat{u}, f)$ opened by A' when \hat{u} arrived. Thus, the total cost of A' for the demands in C is at most $2f$. If $\text{diam}(C) > f$, A' covers the demands in C by opening, in the worst case, a sequence of $\ell + 1$ clusters $C(\hat{u}, f), C(\hat{u}, 2f), \dots, C(\hat{u}, 2^\ell f)$, where $\ell = \lceil \log_2(\text{diam}(C)/f) \rceil$. Thus, the total cost of A' for the demands in C is at most

$$\sum_{i=0}^{\ell} (1 + 2^i)f = (\ell + 2^{1+\ell})f \leq 5 \text{diam}(C),$$

where the inequality follows from $\lceil \log_2 x \rceil \leq x$ and $\lceil \log_2 x \rceil \leq 1 + \log_2 x$, for all $x > 1$. \square

A.3 The Solution Mapping in the Proof of Theorem 1

We first describe an online mapping of any feasible solution of \mathcal{I} to a feasible solution of \mathcal{I}' of equal cost. By the construction of T , a permit of type k that covers the driving days in an interval D_k in \mathcal{I} corresponds to a node v_k at level k of T , in the sense that opening a cluster $C(v_k, c_k - 1)$ covers all demands corresponding to the driving days in D_k . Moreover, the cost of $C(v_{D_k}, c_k - 1)$ is c_k , i.e., equal to the cost of the corresponding permit. Therefore, opening the clusters corresponding to the permits bought by a feasible solution of \mathcal{I} gives a feasible solution of \mathcal{I}' of equal cost.

For the converse mapping, we assume that in any feasible solution of \mathcal{I}' , all clusters are centered at nodes at levels $1, \dots, K$ of T and that any cluster centered at a level- k node v_k has radius $c_k - 1$. This assumption is essentially without loss of generality, since any feasible solution without this property can be translated into a feasible solution of no greater cost that satisfies this property. Indeed, let $C_k = C(v_k, r)$ be any cluster rooted at v_k . If v_k is a leaf, we can root C_k at the ancestor of v_k (recall that a leaf and its ancestor are at distance 0 to each other). If $r < c_k - 1$, C_k does not cover any leaves, and can be safely removed from the solution. Finally, if for some level $j \geq k$, $r \in [c_j - 1, c_{j+1} - 1)$, we can replace C_k by a new cluster which is rooted at the level- j ancestor of v_k and has a radius of $c_j - 1$. The new cluster covers all demand covered by C_k at no greater cost.

In such a solution, each cluster $C(v_k, c_k - 1)$ costs c_k and, by the construction of T , corresponds to a parking permit of type k that covers all the driving days corresponding to the demand points in the subtree rooted at v_k . Therefore, buying the parking permits corresponding to the clusters opened by a feasible solution of \mathcal{I}' gives a feasible solution of \mathcal{I} of equal cost. \square

A.4 The Proof of Lemma 1

Given an instance \mathcal{I} of OnlSumRad in an HST with K levels, we construct an instance \mathcal{I}' of the interval version of ParkPermit with K permits, such that any feasible solution of \mathcal{I} is mapped, in an online fashion, to a feasible solution of \mathcal{I}' of equal cost, and vice versa.

Let \mathcal{I} be an instance of OnlSumRad in an α -HST T with K levels, where all nodes at level k , $1 \leq k \leq K - 1$, have the same number n_k of children, and all demands are located at the leaves of T . Without loss of generality, we assume that the cluster opening cost is $f = 1$. The permits of \mathcal{I}' essentially reflects the structure of T . Specifically, there is a day in the schedule of \mathcal{I}' corresponding to each leaf of T . For each leaf v_0 , there is a permit of type 0 with cost $c_0 = 1$ and duration $d_0 = 1$. This permit covers the day corresponding to v_0 and is equivalent to a cluster $C(v_0, 0)$ of cost 1. Similarly, for each node v_k at level k of T , $1 \leq k \leq K - 1$, there is a permit of type k with cost $c_k = (\alpha^k + \alpha - 2)/(\alpha - 1)$ and duration $d_k = \prod_{j=1}^k n_j$. This permit covers the days corresponding to the leaves of the subtree rooted at v_k and is equivalent to a cluster $C(v_k, (\alpha^k - 1)/(\alpha - 1))$ of cost equal to c_k . The permits of type $k - 1$ corresponding to the children of v_k in T are embedded in the permit of type k corresponding to v_k , in the sense that the intervals covered by the former permits form a partition of the interval covered by the latter. As for the demand sequence of \mathcal{I}' , for each demand of \mathcal{I} located at a leaf v_0 of T , the day corresponding to v_0 in \mathcal{I}' is marked as a driving day⁴.

Next, we describe an online mapping of any feasible solution of \mathcal{I} to a feasible solution of \mathcal{I}' of equal cost. Similarly to the proof of Theorem 1, we assume, without loss of generality, that in any feasible solution of \mathcal{I} , any cluster centered at a level- k node v_k has a radius of $(\alpha^k - 1)/(\alpha - 1)$. Then, each cluster $C(v_k, (\alpha^k - 1)/(\alpha - 1))$ costs c_k , and corresponds to a permit of type k that covers all driving days corresponding to leaves of the subtree rooted at v_k . Therefore, purchasing the permits corresponding to the clusters of a feasible solution of \mathcal{I}' gives a feasible solution of \mathcal{I} of equal cost.

For the converse mapping, we observe that a permit of type k that covers the driving days in an interval D_k corresponds to a level- k node v_k of T , in the sense that opening a cluster $C(v_k, c_k - 1)$, of cost c_k , covers all demand points corresponding to the driving days in D_k . Therefore, opening the clusters corresponding to the permits purchased by a feasible solution of \mathcal{I} gives a feasible solution of \mathcal{I}' of equal cost. \square

⁴ We highlight that the leaves of T can appear in the demand sequence of \mathcal{I} in any order. Thus, we require that the ParkPermit algorithm is time-sequence-independent, i.e., it can handle driving requests that arrive out of the time order.

A.5 The Proof of Theorem 3

We first show that a constant-distortion planar embedding of a ternary α -HST T implies the theorem. Specifically, let α be any constant in $(2, 3)$, and let D_α be the distortion of an embedding e that maps each node v of T to a point $e(v)$ in the plane. Namely, for every pair of nodes u, v of T , we have that $d_T(u, v)/D_\alpha \leq d_P(e(u), e(v)) \leq d_T(u, v)$, where $d_T(u, v)$ (resp. $d_P(u, v)$) denotes the distance of u and v in T (resp. in the Euclidean plane). Assuming the embedding e and a c -competitive deterministic algorithm A for OnSumRad on the plane, we describe a $2cD_\alpha$ -competitive algorithm A' for T .

For any demand point u in T , we present the algorithm A with a demand located at $e(u)$. If A covers $e(u)$ by opening a new cluster $C(v, r)$, the algorithm A' opens a new cluster $C(u, 2D_\alpha r)$. Then, for every node z of T for which $e(z)$ is covered by $C(v, r)$, z is covered by the corresponding cluster $C(u, 2D_\alpha r)$ of A' . This holds because $d_P(e(u), e(v)) \leq 2r$ and the distortion of e is D_α . If $e(u)$ is covered by an existing cluster of A , the previous observation implies that u is covered by the corresponding cluster of A' .

Since for any demand points u, u' , the distance of $e(u)$ and $e(u')$ in the plane is no greater than their distance in T , the optimal cost of the instance presented to A is no greater than the optimal cost of the instance presented to A' . Also, the cost of each cluster of A' is at most $2D_\alpha$ times the cost of the corresponding cluster of A . Therefore, the competitive ratio of A' is at most $2D_\alpha c$. Since D_α is a constant and, by Theorem 2, the competitive ratio of A' is $\Omega(\log n)$, the competitive ratio of A is $\Omega(\log n)$ as well.

To conclude the proof, we describe a D_α -distortion embedding of a ternary α -HST T with $K + 1$ levels in the Euclidean plane. The root of T is mapped to the point $(0, 0)$. The children of the root are mapped to the points $(-\alpha^K, 0)$, $(0, \alpha^K)$, $(\alpha^K, 0)$. For each level- k node v_k , $k = K, \dots, 1$, whose parent is located along the x -axis on the left (resp. on the right), its children are mapped to the 3 points at distance α^{k-1} to v_k located along the x -axis on the right (resp. on the left) and along the y -axis up and down. For each level- k node v_k , $k = K, \dots, 1$, whose parent is located down along the y -axis, its children are mapped to the 3 points at distance α^{k-1} to v_k located up along the y -axis and left and right along the x -axis (see also Fig. 2).

We proceed to show that the distortion of this embedding is at most $\sqrt{2}\alpha/(\alpha-2)$. We first observe that for any two nodes u, v of T , $d_P(e(v), e(u)) \leq d_T(u, v)$, i.e., the distance of u and v in T is no less than the distance of their images $e(u)$ and $e(v)$ in the plane. Moreover, due to the self-similarity of the embedding, the maximum distortion occurs for pairs of leaves of T mapped to points in the plane that lie at symmetric locations with respect to the line $y = x$ (or to the line $y = -x$) and are closest to it (e.g., such are the pairs of leaves/points 21 and 24, 22 and 23, 31 and 32, and 30 and 33 in Fig. 2). The distance of any such a pair of leaves u, v in T is $d_T(u, v) = 2(\alpha^{K+1} - 1)/(\alpha - 1)$. On the other hand, the distance of their images $e(u), e(v)$ in the Euclidean plane is:

$$d_P(e(u), e(v)) = \sqrt{2} \left(\alpha^K - \frac{\alpha^K - 1}{\alpha - 1} \right) = \sqrt{2} \frac{\alpha^{K+1} - 2\alpha^K + 1}{\alpha - 1}$$

Therefore, the maximum distortion of the embedding is:

$$D_\alpha = \frac{2(\alpha^{K+1} - 1)}{\sqrt{2}(\alpha^{K+1} - 2\alpha^K + 1)} \leq \frac{\sqrt{2}\alpha}{\alpha - 2},$$

where the inequality holds for all $\alpha \in (2, 3)$. □

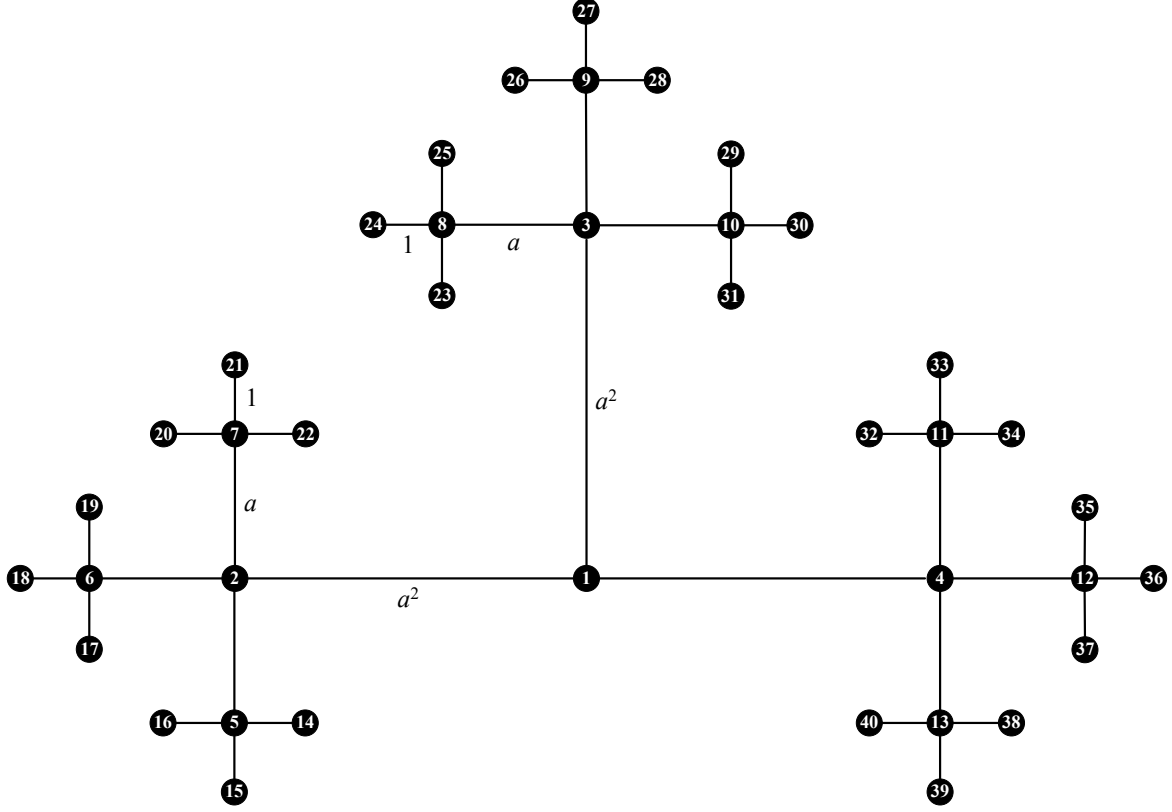


Fig. 2. An example of the embedding used in the proof of Theorem 3. The nodes and the edges connecting them depict the structure of a ternary α -HST with 4 levels and $\alpha \approx 2.5$. The locations of the nodes correspond to the locations in the plane to which they are mapped by the embedding.

A.6 PD-SumRad: Feasibility of the Dual Solution

We show that the dual solution maintained by PD-SumRad satisfies all the dual constraints. In the dual solution maintained by PD-SumRad, each variable a_j is either 0 or f . Since the righthand-side of any constraint is a multiple of f , no constraint can be violated without first becoming tight. To prove the lemma, we show that after a constraint becomes tight, its lefthand-side does not increase, and thus the constraint will never be violated.

We call a cluster $C(z, r_k)$ *tight* if the dual constraint corresponding to (z, k) is satisfied with equality. We next prove that as soon as a cluster $C(z, r_k)$ becomes tight, each subsequent demand $u \in C(z, r_k)$ is covered by some open cluster of PD-SumRad, and thus the corresponding dual variable is set to 0. To this end, let us consider some cluster $C(z, r_k)$ that becomes tight when a demand u_j is processed. Then, $d(u_j, z) \leq r_k$. We assume that to cover u_j , PD-SumRad opens a new cluster $C' = C(z', 3r_{k'})$. The algorithm ensures that $k' \geq k$ (and thus $r_{k'} \geq r_k$) and that $d(u_j, z') \leq r_{k'}$. Now let u be any subsequent demand in $C(z, r_k)$. Since

$$d(u, z') \leq d(u, u_j) + d(u_j, z') \leq 2r_k + r_{k'} \leq 3r_{k'},$$

u is covered by C' . The first inequality above holds because the metric space satisfies the triangle inequality; the second holds because both u and u_j belong to $C(z, r_k)$. Finally, the third inequality follows from $r_{k'} \geq r_k$. \square

A.7 The Proof of Lemma 3

We recall the assumption that the optimal solution only consists of clusters with radius of the form $2^k f$, where k is a non-negative integer. To establish the competitive ratio, we consider an optimal cluster $C(p, 2^k f)$ of total cost $(2^k + 1)f$, $k \leq \log_2 n$, and bound the expected cost of the algorithm until it opens a cluster that includes the entire cluster $C(p, 2^k f)$.

Let u_1, u_2, \dots, u_T be the sequence of demands, where the cluster that u_T opens covers all of $C(p, 2^k f)$. Notice that T itself is a random variable. For each demand u_i , we let X_i be the random variable for the cost of the clusters that u_i opens. Hence, the total algorithm's cost for u_1, u_2, \dots, u_T is $X = \sum_{i=1}^T X_i$. For each demand u_i , X_i is 0 if u_i is covered upon arrival. Otherwise, X_i follows the distribution in the description of Simple-SumRad. Let Y_i be a new random variable such that $Y_i = X_i$ if u_i is not covered, else Y_i takes a value as if u_i was not covered at its arrival time. Clearly, for each i , $X_i \leq Y_i$. Thus, the expected cost of Simple-SumRad until a cluster covering all of $C(p, 2^k f)$ opens is:

$$\mathbb{E} \left[\sum_{i=1}^T X_i \right] \leq \mathbb{E} \left[\sum_{i=1}^T Y_i \right]$$

We observe that Y_i are nonnegative, independent and identically distributed random variables, and that T is a stopping time. Hence, by Wald's equation we have that $\mathbb{E}[\sum_{i=1}^T Y_i] = \mathbb{E}[Y] \cdot \mathbb{E}[T]$, where Y denotes the (identical) distribution of Y_1, \dots, Y_T .

$\mathbb{E}[T]$ denotes the expected number of demands in $C(p, 2^k f)$ that have arrived before the first of them opens a new cluster of radius $2^{k+1} f$ that includes the entire cluster $C(p, 2^k f)$. Hence, $\mathbb{E}[T] = 2^{k+1}$. Moreover, we have that:

$$\mathbb{E}[Y] = \sum_{i=0}^{\log_2 n} \frac{1}{2^i} (2^i + 1)f \leq (3 + \log_2 n)f$$

Taking also into account the cost of $(2^{k+1} + 1)f$ for the cluster of radius $2^{k+1} f$ opened by u_T , the expected cost of the algorithm for the demands in $C(p, 2^k f)$ is at most $(2^{k+1}(3 + \log_2 n) + 2^{k+1} + 1)f$, which is at most $2(4 + \log_2 n)$ times the optimal cost for $C(p, 2^k f)$. \square

A.8 The Proof of Lemma 4

We consider an instance \mathcal{I} of OnlSumRad in the d -dimensional Euclidean space. We assume, without loss of generality, that the cluster opening cost is $f = 1$ and that the demands lie in a bounded d -dimensional hypercube of side length Δ . Moreover, we assume that the number of demands n is an integral power of 2 and given in advance. Given \mathcal{I} , we construct an instance \mathcal{I}' of OnlSumRad in a 2-HST T with $O(\log n)$ levels, where all nodes at the same level have the same number of children, and all n demands are located at the leaves of T . The construction ensures that (a) any feasible solution of \mathcal{I} of cost c is mapped to a feasible solution of \mathcal{I}' of cost at most $O(2^d c)$; and (b) any feasible solution of \mathcal{I}' of cost c' is mapped, in an online fashion, to a feasible solution of \mathcal{I} of cost at most $\sqrt{d} c'$. Consequently, given a $c(K)$ -competitive algorithm for \mathcal{I}' , we can apply this transformation and obtain a $O(2^d \sqrt{d} c(K))$ -competitive algorithm for \mathcal{I} , where $K = O(\log n)$.

The 2-HST T that determines the metric space of \mathcal{I}' corresponds to a hierarchical decomposition \mathcal{D} of the d -dimensional space into hypercubes with $K + 1$ levels, where $K = \log_2 n$. More precisely, the highest level K of the decomposition partitions the space into $\lceil \Delta/n \rceil^d$ level- K d -dimensional hypercubes, each with a side length of $2^K = n$. Each level- k hypercube, $k = K, \dots, 1$, has a side length of 2^k , and is partitioned into 2^d level- $(k - 1)$ children hypercubes of side length 2^{k-1} .

The root of the 2-HST T (at level $K + 1$) has as many children as the number of level- K hypercubes. The distance between the root and the children is set to some arbitrarily large number (to comply with the definition of HSTs, one may replace the edges from the root to its children by appropriately long paths, where the edge length drops by a factor of 2 on each level). There is an HST node v_k at level k , $k = K, \dots, 1$, corresponding to each level- k hypercube h_k in \mathcal{D} . Each level- k node v_k has 2^d children in T that correspond to the 2^d children of h_k in \mathcal{D} . The distance in T between each level- k node to its children is 2^{k-1} . The level-0 nodes of T are leaves and do not have any children. The cluster opening cost is $f = 1$. For each demand u_j in \mathcal{I} , there is a demand \hat{u}_j located at the leaf of T corresponding to the level-0 hypercube that includes u_j in the hierarchical decomposition \mathcal{D} .

To establish property (a) above, we consider a feasible solution of \mathcal{I} , and let $C(v, 2^{k-1})$, with $1 \leq k \leq K + 1$, be any of its clusters. We observe that $C(v, 2^{k-1})$ intersects at most 2^d level- k hypercubes $h_k^1, \dots, h_k^{2^d}$ of the hierarchical decomposition \mathcal{D} . Hence, all demands in T that correspond to the demands in \mathcal{I} covered $C(v, 2^{k-1})$ can be covered by 2^d clusters with cost 2^k rooted at the level- k nodes of T that correspond to $h_k^1, \dots, h_k^{2^d}$. The total cost of these clusters in T is $2^d 2^k$, i.e., at most 2^{d+1} times the total cost of $C(v, 2^{k-1})$. Therefore, any feasible solution of \mathcal{I} of cost c can be mapped to a feasible solution of \mathcal{I}' of cost at most $O(2^d c)$.

As for property (b), we consider any feasible solution of \mathcal{I}' with total cost c . As in the proof of Theorem 1, we assume, without loss of generality, that all clusters are centered at nodes at levels $1, \dots, K$, and that any cluster centered at a level- k node v_k has a radius of $2^k - 1$. Let $C(v_k, 2^k - 1)$ be any cluster centered at a level- k node v_k of T . Then, the demands in \mathcal{I} that correspond to the demands covered by $C(v_k, 2^k - 1)$ in \mathcal{I}' belong to a single level- k hypercube h_k of the hierarchical decomposition \mathcal{D} . Hence, all these demands can be covered by a single cluster with radius $\sqrt{d} \cdot 2^{k-1}$ centered at the center of h_k . Therefore, given a feasible solution S of \mathcal{I}' of cost c , opening the clusters in the d -dimensional Euclidean space that correspond to the clusters of S , we obtain a feasible solution of \mathcal{I} of cost at most $\sqrt{d} c$. \square

A.9 Frac-SumRad: Estimating the Number of Demands

To remove the assumption that n is known to Frac-SumRad in advance, we run Frac-SumRad in phases, where each phase ℓ uses an estimation $n_\ell = 2^{2^{\ell}}$ of n . Phase ℓ , where $\ell = 1, 2, \dots$, ends just after its processing of n_ℓ demands. Then, the algorithm keeps the fractional solution for the demands arriving in phase ℓ , and starts computing a new fractional solution for the next demands arriving in phase $\ell + 1$, with an estimation $n_{\ell+1}$ of n .

We show that running Frac-SumRad in phases increases its competitive ratio by no more than a constant factor. Let λ be the last phase of Frac-SumRad. By Theorem 6, the cost of Frac-SumRad in phase ℓ , $\ell = 1, \dots, \lambda$, is at most $2^\ell \beta \text{OPT}_\ell$, where OPT_ℓ is the optimal cost for the demands arriving in phase ℓ , and β is the constant hidden in the O -notation, in Theorem 6. Since the optimal cost OPT for all demands is no less than OPT_ℓ , the total cost of Frac-SumRad is at most $2^{\lambda+1} \beta \text{OPT}$. On the other hand, the total number of demands is at least $2^{2^{\lambda-1}}$, because the phase $\lambda - 1$ is complete, and $\log \log n \geq 2^{\lambda-1}$. Therefore, the total cost of Frac-SumRad is at most $4\beta 2^{\lambda-1} \text{OPT}$, and the competitive ratio is $O(\log \log n)$.